**WEEK 2: Mockito Hands-On Exercises**

**Exercise 1:** Mocking and Stubbing

**Scenario:** You need to test a service that depends on an external API. Use Mockito to mock the external API and stub its methods.

**Code:**

**ExternalApi.java**

public interface ExternalApi {

    String getData();

}

**MyService.java**

public class MyService {

    private final ExternalApi externalApi;

    public MyService(ExternalApi externalApi) {

        this.externalApi = externalApi;

    }

    public String fetchData() {

        return externalApi.getData();

    }

}

**MyServiceTest.java**

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

    @Test

    public void testExternalApi() {

        ExternalApi mockApi = mock(ExternalApi.class);

        when(mockApi.getData()).thenReturn("Mock Data");

        MyService service = new MyService(mockApi);

        String result = service.fetchData();

        assertEquals("Mock Data", result);

    }

}

**Output:**



**Exercise 2:** Verifying Interactions

**Scenario:** You need to ensure that a method is called with specific arguments.

**Code:**

**ApiClient.java**

public interface ApiClient {

    String getData();

}

**DataService.java**

public class DataService {

    private final ApiClient apiClient;

    public DataService(ApiClient apiClient) {

        this.apiClient = apiClient;

    }

    public void fetch() {

        apiClient.getData();

    }

}

**DataServiceTest.java**

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class DataServiceTest {

    @Test

    public void testFetchCallsApiClient() {

        ApiClient mockClient = mock(ApiClient.class);

        DataService service = new DataService(mockClient);

        service.fetch();

        verify(mockClient).getData();  // Verifies method was called

    }

}

**Output:**

